

DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUU	UUU	GGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUU	UUU	GGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUU	UUU	GGGGGGGG

[illegible]

```
1 0001 0 MODULE DBGNHELP (IDENT = 'V04-000') =
2 0002 0
3 0003 1 BEGIN
4 0004 1
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 *   COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 *   DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 *   ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 *   THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 *   ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 *   INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 *   COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 *   OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 *   TRANSFERRED.
18 0018 1 *
19 0019 1 *   THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 *   AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 *   CORPORATION.
22 0022 1 *
23 0023 1 *   DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 *   SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 MODULE FUNCTION
31 0031 1   This module contains the ATN parse and command execution networks to
32 0032 1   support the HELP command. The routine DBG$NEXECUTE_HELP consists of
33 0033 1   the version 2 debugger routine DBG$GET_HELP with a few modifications
34 0034 1   to allow it to perform correctly in version 3. Specifically, error
35 0035 1   signals (which unwind the stack) have been replaced with calls to
36 0036 1   version 3 error output routines.
37 0037 1
38 0038 1
39 0039 1 AUTHOR:      David Plummer, CREATION DATE:  4/9/80
40 0040 1
41 0041 1 MODIFIED BY:
42 0042 1
43 0043 1      Richard Title   15 Dec 1981   Converted to the new help
44 0044 1      librarian LBR$OUTPUT_HELP,
45 0045 1      which prompts for output
46 0046 1      ("Topic? ", "Subtopic? ")
47 0047 1      Richard Title   13-Jun 1982   Added support for DBG$HELP
48 0048 1      (logical name telling where
49 0049 1      the help library is)
50 0050 1
51 0051 1
52 0052 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
53 0186 1
54 0187 1 LIBRARY 'LIB$:DBGGEN.L32';
55 0188 1
56 0189 1 FORWARD ROUTINE
57 0190 1   DBG$NPARSE_HELP,           ! Creates the command execution tree for help
```

DBGNHELP  
V04-000

I 10  
16-Sep-1984 01:46:06 VAX-11 Bliss-32 V4.0-742  
14-Sep-1984 12:17:16 [DEBUG.SRC]DBGNHELP.B32;1

Page 2  
(1)

:	58	0191	1	DBG\$NEXECUTE_HELP,
:	59	0192	1	PRINT_HELP_LINE,
:	60	0193	1	INPUT_HELP_LINE;

! Executes the parsed HELP command  
! Print a line of HELP text  
! Input HELP topic or subtopic request

```
62 0194 1 EXTERNAL ROUTINE
63 0195 1   DBGSGET_TEMPMEM,
64 0196 1   DBGSNEWLINE: NOVALUE,
65 0197 1   DBGSNMAKE_ARG_VECT,
66 0198 1   DBGSPRINT: NOVALUE,
67 0199 1   DBGS$SCR_OUTPUT_SCREEN: NOVALUE,
68 0200 1   DBGS$SCR_SCREEN_MODE: NOVALUE,
69 0201 1   LBR$INI_CONTROL,
70 0202 1   LBR$OPEN,
71 0203 1   LBR$CLOSE,
72 0204 1   LBR$GET_HELP,
73 0205 1   LBR$OUTPUT_HELP,
74 0206 1   LIB$PUT_OUTPUT,
75 0207 1   LIB$GET_INPUT,
76 0208 1   SMGSREAD_COMPOSED_LINE,
77 0209 1   SYS$TRNLOG;
78 0210 1
79 0211 1 EXTERNAL
80 0212 1   DBGS$CHAR_TABLE: VECTOR[.BYTE],
81 0213 1   DBGS$GB_KEYPAD_INPUT: BYTE,
82 0214 1   DBGS$GL_CISHEAD: REF CISS$LINK,
83 0215 1   DBGS$GL_INPRAB: BLOCK[.BYTE],
84 0216 1   DBGS$GL_KEYBOARD_ID,
85 0217 1   DBGS$GL_KEY_TABLE_ID,
86 0218 1   DBGS$GL_OUTPRAB: BLOCK[.BYTE],
87 0219 1   DBGS$GL_SCREEN_MODE;
88 0220 1
89 0221 1 EXTERNAL LITERAL
90 0222 1   SMGS_EOF;
```

```
! Allocates temporary dynamic storage
! Go to a new print line
! Constructs a message argument vector
! Print some debug output
! Output the current screen contents
! Turn screen mode on or off
! Librarian init control table
! Librarian open library file
! Librarian close library file
! Librarian get help
! Librarian output help
! Library output routine
! Library input routine
! Read a line of input in keypad mode
! Translate logical name

! Character type table
! Flag set if keypad input is active
! Head of Command Input Stream
! RAB for DEBUG input (DBGS$INPUT)
! The keyboard Id used for keypad input
! The key-table Id used for keypad input
! RAB for DEBUG output (DBGS$OUTPUT)
! Flag set if screen mode is active

! Keypad input End-of-File code
```

```

92 0223 1 GLOBAL ROUTINE DBG$NPARSE_HELP (INPUT_DESC, VERB_NODE, MESSAGE_VECT) =
93 0224 1
94 0225 1 FUNCTION
95 0226 1     DBG$NPARSE_HELP constructs the command execution tree for the HELP
96 0227 1     command. Specifically, a Noun Node is allocated and linked to the
97 0228 1     Verb Node. A copy of the present input descriptor (describing the
98 0229 1     input line minus the keyword HELP) is made, and the value of the Noun
99 0230 1     Node is a pointer to this descriptor. The actual parsing of the HELP
100 0231 1     string is done by DBG$NEXECUTE_HELP.
101 0232 1
102 0233 1 INPUTS
103 0234 1     INPUT_DESC - Descriptor of the present input line.
104 0235 1
105 0236 1     VERB_NODE - Head of the command execution tree.
106 0237 1
107 0238 1     MESSAGE_VECT - The address of a longword to contain the address
108 0239 1     of a message argument vector.
109 0240 1
110 0241 1 OUTPUTS
111 0242 1     INPUT_DESC - The input line string descriptor is updated to reflect
112 0243 1     the current parse location. This normally means that the
113 0244 1     input descriptor winds up pointing past the end of the line.
114 0245 1
115 0246 1     VERB_NODE - The Verb Node is filled in to reflect the parameters
116 0247 1     parsed on the HELP command.
117 0248 1
118 0249 1     An unsigned integer longword completion code is returned as the
119 0250 1     routine's value. These are the possible values:
120 0251 1
121 0252 1         ST$K_SEVERE (4) - The input could not be parsed.
122 0253 1         ST$K_SUCCESS (1) - The input was parsed and an execution
123 0254 1         tree was created.
124 0255 1
125 0256 1
126 0257 1 BEGIN
127 0258 1
128 0259 1 MAP
129 0260 1     INPUT_DESC: REF DBG$STG_DESC, ! Pointer to input string descriptor
130 0261 1     VERB_NODE: REF DBG$VERB_NODE; ! Pointer to the Verb Node
131 0262 1
132 0263 1 LOCAL
133 0264 1     NOUN_NODE: REF DBG$NOUN_NODE, ! Pointer to Noun node for execution tree
134 0265 1     COMMAND_DESC: REF DBG$STG_DESC; ! Descriptor of HELP line
135 0266 1
136 0267 1
137 0268 1 ! Get storage for the Noun Node and link it to the Verb Node.
138 0269 1 !
139 0270 1 NOUN_NODE = DBG$GET_TEMPMEM (DBG$K_NOUN_NODE_SIZE);
140 0271 1 VERB_NODE [DBG$L_VERB_OBJECT_PTR] = .NOUN_NODE;
141 0272 1
142 0273 1
143 0274 1 ! Get storage for the command descriptor.
144 0275 1 !
145 0276 1 COMMAND_DESC = DBG$GET_TEMPMEM (2);
146 0277 1
147 0278 1
148 0279 1
```

```

: 149      0280      2      ! Copy the fields of the input descriptor to the command descriptor.
: 150      0281      2
: 151      0282      2      !
: 152      0283      2      ! COMMAND_DESC [DSC$W_LENGTH] = .INPUT_DESC [DSC$W_LENGTH];
: 153      0284      2      ! COMMAND_DESC [DSC$A_POINTER] = .INPUT_DESC [DSC$A_POINTER];
: 154      0285      2
: 155      0286      2
: 156      0287      2      ! Set the value of the Noun Node to the address of the command descriptor.
: 157      0288      2      !
: 158      0289      2      ! NOUN_NODE [DBG$L_NOUN_VALUE] = .COMMAND_DESC;
: 159      0290      2
: 160      0291      2
: 161      0292      2      ! Eat the rest of the input command and return.
: 162      0293      2      !
: 163      0294      2      ! INPUT_DESC [DSC$W_LENGTH] = 0;
: 164      0295      2      ! RETURN ST$K_SUCCESS;
: 165      0296      2      !
:                               END;
```

```

.TITLE  DBGNHELP
.IDENT  \V04-000\

.EXTRN  DBG$GET_TEMPMEM
.EXTRN  DBG$NEWLINE, DBG$MAKE_ARG_VECT
.EXTRN  DBG$PRINT, DBG$SCR_OUTPUT_SCREEN
.EXTRN  DBG$SCR_SCREEN_MODE
.EXTRN  LBR$INI_CONTROL
.EXTRN  LBR$OPEN, LBR$CLOSE
.EXTRN  LBR$GET_HELP, LBR$OUTPUT_HELP
.EXTRN  LIB$PUT_OUTPUT, LIB$GET_INPUT
.EXTRN  SMG$READ_COMPOSED_LINE
.EXTRN  SYS$TRNLOG, DBG$CHAR_TABLE
.EXTRN  DBG$GB_KEYPAD_INPUT
.EXTRN  DBG$GL_CISHEAD, DBG$GL_INPRAB
.EXTRN  DBG$GL_KEYBOARD_ID
.EXTRN  DBG$GL_KEY_TABLE_ID
.EXTRN  DBG$GL_OUTPRAB, DBG$GL_SCREEN_MODE
.EXTRN  SMG$_EOF
```

```
.PSECT  DBG$CODE, NOWRT, SHR, PIC, 0
```

			000C 00000	.ENTRY  DBG\$NPARSE_HELP, Save R2,R3	: 0223
	53	00000000G	00 9E 00002	MOVAB  DBG\$GET_TEMPMEM, R3	: 0271
			04 DD 00009	PUSHL  #4	: 0272
	63		01 FB 0000B	CALLS  #1, DBG\$GET_TEMPMEM	: 0277
	52		50 D0 0000E	MOVL   R0, NOUN_NODE	: 0282
08	50	08	AC D0 00011	MOVL   VERB_NODE, R0	: 0283
	A0		52 D0 00015	MOVL   NOUN_NODE, 8(R0)	: 0288
			02 DD 00019	PUSHL  #2	: 0293
	63		01 FB 0001B	CALLS  #1, DBG\$GET_TEMPMEM	: 0294
	51	04	AC D0 0001E	MOVL   INPUT_DESC, R1	: 0296
	60		61 B0 00022	MOVW   (R1), -(COMMAND_DESC)	
04	A0	04	A1 D0 00025	MOVL   4(R1), 4(COMMAND_DESC)	
	62		50 D0 0002A	MOVL   COMMAND_DESC, (NOUN_NODE)	
			61 B4 0002D	CLRW   (R1)	
	50		01 D0 0002F	MOVL   #1, R0	
			04 00032	RET	

DBGNHLP  
V04-000

M 10  
16-Sep-1984 01:46:06  
14-Sep-1984 12:17:16

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGNHLP.B32;1

Page 6  
(3)

; Routine Size: 51 bytes,      Routine Base: DBG\$CODE + 0000

```
167 0297 1 GLOBAL ROUTINE DBG$NEXECUTE_HELP (VERB_NODE) =
168 0298 1
169 0299 1 FUNCTION
170 0300 1     Invoke the VMS librarian to implement the HELP command.
171 0301 1
172 0302 1 INPUTS
173 0303 1     VERB_NODE - A pointer to the Verb Node that forms the head of the
174 0304 1     command execution tree for the HELP command.
175 0305 1
176 0306 1 OUTPUTS
177 0307 1     This routine always returns STS$K_SUCCESS as its value.
178 0308 1
179 0309 1
180 0310 2 BEGIN
181 0311 2
182 0312 2 MAP
183 0313 2     VERB_NODE: REF DBG$VERB_NODE;    ! Pointer to the input Verb Node
184 0314 2
185 0315 2 LOCAL
186 0316 2     CHAR,                ! Temporary placeholder for a char
187 0317 2     COUNT,             ! Counter for leading blanks
188 0318 2     DBGHELP_STGDESC: BLOCK[8,BYTE], ! String descriptor for DBG$HELP
189 0319 2     DBGHELP_STG: VECTOR[8,BYTE],       ! String with DBG$HELP
190 0320 2     DUMMY: VECTOR [2],              ! Output string descriptor for
191 0321 2                                     SYS$TRNLOG
192 0322 2     DUMMY_BUFFER: VECTOR [256, BYTE], ! Output buffer for SYS$TRNLOG
193 0323 2     INPUT_PTR,             ! Temporary pointer into input
194 0324 2     LIB_NAME: REF DBG$STG_DESC, ! descriptor for library name
195 0325 2     NOUN_NODE: REF DBG$NOUN_NODE, ! noun node of command execution tree
196 0326 2     PARSE_STG_DESC: REF DBG$STG_DESC, ! Descriptor of the help command
197 0327 2     SAVED_PARSE_STG_DESC,      ! ???
198 0328 2     SCREEN_MODE_FLAG,         ! Saved value of screen mode flag
199 0329 2     STATUS,                   ! Librarian routines return status
200 0330 2     TRNLOG;                  ! Return status from $TRNLOG
201 0331 2
202 0332 2
203 0333 2
204 0334 2 ! Recover pointers to the Noun Node and the command descriptor.
205 0335 2
206 0336 2 NOUN_NODE = .VERB_NODE [DBG$L_VERB_OBJECT_PTR];
207 0337 2 PARSE_STG_DESC = .NOUN_NODE [DBG$L_NOUN_VALUE];
208 0338 2
209 0339 2
210 0340 2 ! Initialize the library name. If the logical name DBG$HELP is defined,
211 0341 2 ! then we use that as the directory in which to find DEBUGHLP.HLB.
212 0342 2 ! Otherwise, the help librarian looks in SYS$HELP by default.
213 0343 2
214 0344 2 LIB_NAME = DBG$GET_TEMPMEM (2);
215 0345 2 DUMMY[0] = %X'010E0000' + 256;
216 0346 2 DUMMY[1] = DUMMY_BUFFER;
217 0347 2
218 0348 2
219 0349 2 ! Set up string descriptor for DBG$HELP. Use DBG$HELP as the directory
220 0350 2 ! name if a logical name translation exists for DBG$HELP.
221 0351 2
222 0352 2 DBGHELP_STGDESC[DSC$B_CLASS] = DSC$K_CLASS_S;
223 0353 2 DBGHELP_STGDESC[DSC$B_DTYPE] = DSC$K_DTYPE_T;
```

```
DBGHELP_STGDESC[DSCSW_LENGTH] = 8;
DBGHELP_STGDESC[DSCSA_POINTER] = DBGHELP_STG;
CH$MOVE(8, UPLIT BYTE (%ASCII 'DBG$HELP'), DBGHELP_STG);
TRNLOG = SYS$TRNLOG (DBGHELP_STGDESC, 0, DUMMY, 0, 0, 0);
IF .TRNLOG EQL $$$_NORMAL
THEN
    BEGIN
        LIB_NAME [DSCSW_LENGTH] = 17;
        LIB_NAME [DSCSA_POINTER] = UPLIT BYTE (%ASCII 'DBG$HELP:DEBUGHLP');
    END
ELSE
    BEGIN
        LIB_NAME [DSCSW_LENGTH] = 8;
        LIB_NAME [DSCSA_POINTER] = UPLIT BYTE(%ASCII 'DEBUGHLP');
    END;

! Suppress leading blanks and tabs.
INPUT_PTR = CH$PTR(.PARSE_STG_DESC[DSCSA_POINTER]);
CHAR = CH$RCHAR(INPUT_PTR);
COUNT = 0;
WHILE .DBG$CHAR_TABLE[.CHAR] EQL 4 DO
    BEGIN
        CHAR = CH$RCHAR_A(INPUT_PTR);
        COUNT = .COUNT + 1;
    END;

! Update the string descriptor to point to the first non-blank character.
PARSE_STG_DESC[DSCSW_LENGTH] = .PARSE_STG_DESC[DSCSW_LENGTH] - .COUNT;
PARSE_STG_DESC[DSCSA_POINTER] = .INPUT_PTR;

! Save away PARSE_STG_DESC before we clobber it.
SAVED_PARSE_STG_DESC = .PARSE_STG_DESC;

! Remove the trailing carriage return from PARSE_STG_DESC.
INPUT_PTR = CH$PTR (.PARSE_STG_DESC[DSCSA_POINTER]);
INPUT_PTR = CH$PLUS (.INPUT_PTR, .PARSE_STG_DESC[DSCSW_LENGTH] - 1);
IF CH$RCHAR(.INPUT_PTR) EQL DBG$K_CAR_RETURN
THEN
    PARSE_STG_DESC[DSCSW_LENGTH] = .PARSE_STG_DESC[DSCSW_LENGTH] - 1;

! Check for all blanks. If so, put zero in PARSE_STG_DESC to tell the
! HELP librarian that no keys were specified.
IF .PARSE_STG_DESC[DSCSW_LENGTH] EQL 0
THEN
    PARSE_STG_DESC = 0;
```

```

! If screen mode is set, turn off screen mode for the duration of the
! HELP command. We restore screen mode after the HELP command completes
! if it was set when the HELP command was entered.
!
SCREEN_MODE_FLAG = .DBG$GL_SCREEN_MODE;
IF .DBG$GL_SCREEN_MODE THEN DBG$SCR_SCREEN_MODE(FALSE);

! Call the library routine to output help text. Note that we restore
! screen mode if appropriate before we signal any error message.
!
STATUS = LBR$OUTPUT_HELP (PRINT_HELP_LINE, 0, .PARSE_STG_DESC,
                          .LIB_NAME, OPLIT$HLP$M_PROMPT), INPUT_HELP_LINE);
IF .SCREEN_MODE_FLAG THEN DBG$SCR_SCREEN_MODE(TRUE);
IF NOT .STATUS THEN SIGNAL(DBG$NOSUCHHELP, 0, .STATUS);

! The HELP has been displayed. Now cleanup and return.
!
PARSE_STG_DESC = .SAVED_PARSE_STG_DESC;
PARSE_STG_DESC[DSC$A_POINTER] = CR$PLUS(.PARSE_STG_DESC[DSC$A_POINTER],
                                         .PARSE_STG_DESC[DSC$W_LENGTH]);
PARSE_STG_DESC[DSC$W_LENGTH] = 0;
RETURN ST$SK_SUCCESS;

END;
```

										.PSECT	DBG\$PLIT,NOWRT, SHR, PIC,0								
48	47	55	42	45	44	3A	50	4C	45	48	24	47	42	44	00000	P.AAA:	.ASCII	\DBG\$HELP\	:
							50	4C	45	48	24	47	42	44	00008	P.AAB:	.ASCII	\DBG\$HELP:DEBUGHLP\	:
													50	4C	00017			:	
							50	4C	48	47	55	42	45	44	00019	P.AAC:	.ASCII	\DEBUGHLP\	:
															00021		.BLKB	3	:
														00000001	00024	P.AAD:	.LONG	1	:
										.PSECT	DBG\$CODE,NOWRT, SHR, PIC,0								
											03FC	00000							
							59	00000000G	00	9E	00002					MOVAB	DBG\$SCR_SCREEN_MODE, R9	:	0297
							58	00000000'	EF	9E	00009					MOVAB	P.AAA, R8	:	
							5E	FEE8	CE	9E	00010					MOVAB	-280(SP), SP	:	
							50	04	AC	D0	00015					MOVL	VERB_NODE, R0	:	0336
							50	08	A0	D0	00019					MOVL	8(ROT, NOUN_NODE	:	
							57		60	D0	0001D					MOVL	(NOUN_NODE), PARSE_STG_DESC	:	0337
									02	DD	00020					PUSHL	#2	:	0344
							00000000G	00	01	FB	00022					CALLS	#1, DBG\$GET_TEMPHEM	:	
							56		50	D0	00029					MOVL	R0, LIB_NAME	:	
							E8	AD	010E0100	8F	D0	0002C				MOVL	#17694976, DUMMY	:	0345
							EC	AD		6E	9E	00034				MOVAB	DUMMY_BUFFER, DUMMY+4	:	0346
							F8	AD	010E0008	8F	D0	00038				MOVL	#17694728, DBGHELP_STGDESC	:	0354

FO	AD	FC	AD	FD	AD	9E	00040	MOVAB	DBGHELP_STG, DBGHELP_STGDESC+4	0355
			68		08	28	00045	MOVBC3	#8, P.AXA, DBGHELP_STG	0356
					7E	7C	0004A	CLRQ	-(SP)	0357
					7E	D4	0004C	CLRL	-(SP)	
				E8	AD	9F	0004E	PUSHAB	DUMMY	
					7E	D4	00051	CLRL	-(SP)	
				F8	AD	9F	00053	PUSHAB	DBGHELP_STGDESC	
00000000G	00				06	FB	00056	CALLS	#6, SYS\$TRNLOG	
	01				50	D1	0005D	CMPL	TRNLOG, #1	0358
					0A	12	00060	BNEQ	1\$	
	66				11	B0	00062	MOVW	#17, (LIB_NAME)	0361
04	A6			08	A8	9E	00065	MOVAB	P.AAB, 4(LIB_NAME)	0362
					08	11	0006A	BRB	2\$	0358
	66				08	B0	0006C	MOVW	#8, (LIB_NAME)	0367
04	A6			19	A8	9E	0006F	MOVAB	P.AAC, 4(LIB_NAME)	0368
	20			04	A7	D0	00074	MOVL	4(PARSE_STG_DESC), INPUT_PTR	0374
	51				50	9A	00078	MOVZBL	INPUT_PTR, CHAR	0375
					52	D4	0007B	CLRL	COUNT	0376
04	00000000G	00			41	91	0007D	CMPB	DBG\$CHAR_TABLE[CHAR], #4	0377
					07	12	00085	BNEQ	4\$	
	51				80	9A	00087	MOVZBL	(INPUT_PTR)+, CHAR	0379
					52	D6	0008A	INCL	COUNT	0380
					EF	11	0008C	BRB	3\$	0377
04	67				52	A2	0008E	SUBW2	COUNT, (PARSE_STG_DESC)	0386
	A7				50	D0	00091	MOVL	INPUT_PTR, 4(PARSE_STG_DESC)	0387
	54				57	D0	00095	MOVL	PARSE_STG_DESC, SAVED_PARSE_STG_DESC	0392
	50			04	A7	D0	00098	MOVL	4(PARSE_STG_DESC), INPUT_PTR	0397
	51				67	3C	0009C	MOVZWL	(PARSE_STG_DESC), R1	0398
	51				50	C0	0009F	ADDL2	INPUT_PTR, R1	
	50			FF	A1	9E	000A2	MOVAB	-1(R1), INPUT_PTR	
	0D				60	91	000A6	CMPB	(INPUT_PTR), #13	0399
					02	12	000A'	BNEQ	5\$	
					67	B7	000AB	DECW	(PARSE_STG_DESC)	0401
					67	B5	000AD	TSTW	(PARSE_STG_DESC)	0407
					02	12	000AF	BNEQ	6\$	
					57	D4	000B1	CLRL	PARSE_STG_DESC	0409
	50				00	D0	000B3	MOVL	DBG\$G[SCREEN_MODE], R0	0416
	53				50	D0	000BA	MOVL	R0, SCREEN_MODE_FLAG	
	05				50	E9	000BD	BLBC	R0, 7\$	0417
					7E	D4	000C0	CLRL	-(SP)	
	69				01	FB	000C2	CALLS	#1, DBG\$SCR_SCREEN_MODE	
				0000V	CF	9F	000C5	PUSHAB	INPUT_HELP_LINE	0423
				24	A8	9F	000C9	PUSHAB	P.AAD	0424
					56	DD	000CC	PUSHL	LIB_NAME	
					57	DD	000CE	PUSHL	PARSE_STG_DESC	0423
					7E	D4	000D0	CLRL	-(SP)	
		</								

DBGNHLP  
V04-000

E 11  
16-Sep-1984 01:46:06  
14-Sep-1984 12:17:16

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGNHLP.B32;1

Page 11  
(4)

	57	54	D0 000FC	98:	MOVL	SAVED_PARSE_STG_DESC, PARSE_STG_DESC	:	0431
	50	67	3C 000FF		MOVZWL	(PARSE_STG_DESC), RO	:	0433
04	A7	50	C0 00102		ADDL2	RO, 4(PARSE_STG_DESC)	:	
		67	B4 00106		CLRW	(PARSE_STG_DESC)	:	0434
	50	01	D0 00108		MOVL	#1, RO	:	0435
		04	0010B		RET		:	0437

; Routine Size: 268 bytes, Routine Base: DBG\$CODE + 0033

```

0438 1 ROUTINE PRINT_HELP_LINE(LINE_DESC) =
0439
0440 FUNCTION
0441     Print a line of HELP text to the DEBUG output device. It is necessary
0442     to pass this routine to LBR$OUTPUT_HELP, instead of using the default
0443     routine LIB$PUT_OUTPUT, because DEBUG may write its output to a log
0444     file, to logical name DBG$OUTPUT, or to a screen display, and not
0445     necessarily to SYS$OUTPUT.
0446
0447 INPUTS
0448     LINEDESC - A pointer to a string descriptor for the line to be output.
0449
0450 OUTPUTS
0451     This routine always returns SS$_NORMAL as its value.
0452
0453
0454 BEGIN
0455
0456 MAP
0457     LINE_DESC: REF DBG$STG_DESC;      ! Pointer to output line string descr.
0458
0459
0460
0461     ! Output the line of HELP text via DBG$PRINT. Then return.
0462
0463 $ABORT ON CONTROL Y;
0464 DBG$PRINT(UPLOT BYTE(XASCII '!AD'),
0465           .LINE_DESC[DSC$W_LENGTH], .LINE_DESC[DSC$A_POINTER]);
0466
0467 DBG$NEWLINE();
0468 RETURN SS$_NORMAL;
0469
0470 END;

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
44 41 21 03 00028 P.AAE: .ASCII <3>\!AD\
.EXTRN DBG$GV_CONTROL
.PSECT DBG$CODE,NOWRT, SHR, PIC,0
0000 00000 PRINT_HELP_LINE:
0D 00000000G 00 000280E8 01 E1 00002 .WORD Save nothing : 0438
8F DD 0000A BBC #1, DBG$GV_CONTROL+1, 1$ : 0457
00000000G 00 01 FB 00010 PUSHL #164072
50 04 AC D0 00017 1$: CALLS #1, LIB$SIGNAL
04 A0 DD 0001B MOVL LINE_DESC, R0 : 0465
7E 60 3C 0001E PUSHL 4(R0)
00000000G 00 EF 9F 00021 MOVZWL (R0), -(SP)
00000000G 00 03 FB 00027 PUSHAB P.AAE : 0464
00 00 FB 0002E CALLS #3, DBG$PRINT
50 01 D0 00035 CALLS #0, DBG$NEWLINE : 0466
04 00038 MOVL #1, R0 : 0467
RET : 0469

```

DBGNHLP  
V04-000

G 11  
16-Sep-1984 01:46:06  
14-Sep-1984 12:17:16

VAX-11 Bliss-32 V4.0-742  
[DEBUG.SRC]DBGNHLP.B32;1

Page 13  
(5)

; Routine Size: 57 bytes,      Routine Base: DBG\$CODE + 013F

```
0470 1 ROUTINE INPUT_HELP_LINE (GET_STR, PROMPT_STR) =
0471 1
0472 1 FUNCTION
0473 1 Reads a line of HELP input from the DEBUG input stream. This routine
0474 1 is used by LBR$OUTPUT_HELP to collect responses to the "Topic?" and
0475 1 "Subtopic?" prompts. It is necessary to use this instead of the
0476 1 default LIB$GET_INPUT, because DEBUG may read its input from DBG$INPUT,
0477 1 the keypad read routine, or from an indirect command file, and not
0478 1 necessarily from SYSS$INPUT.
0479 1
0480 1 INPUTS
0481 1 GET_STR - The address of a string descriptor pointing to the buffer
0482 1 to receive the input line.
0483 1
0484 1 PROMPT_STR - A string descriptor specifying the prompt string.
0485 1
0486 1 OUTPUTS
0487 1 GET_STR - The actual length of the read input string is returned to the
0488 1 length field of the GET_STR string descriptor.
0489 1
0490 1 The status returned by the $GET call is returned as this routine's
0491 1 value. If $GET was not called, SSS_NORMAL or the keypad
0492 1 input routine's status is returned.
0493 1
0494 1
0495 2 BEGIN
0496 2
0497 2 MAP
0498 2 GET_STR: REF DBG$STG_DESC, ! String descriptor for input buffer
0499 2 PROMPT_STR: REF DBG$STG_DESC; ! Prompt string string descriptor
0500 2
0501 2 LOCAL
0502 2 CIS_PTR: REF CISC$LINK, ! Pointer to Command Input Stream entry
0503 2 LENGTH, ! The input length on a keypad read
0504 2 STATUS; ! The RMS status code
0505 2
0506 2
0507 2
0508 2 ! If we are currently reading from a DEBUG command list as on an IF, FOR,
0509 2 or WHILE statement, or a screen display DEBUG command list, we simply
0510 2 return a null line to the HELP librarian. This means that HELP commands
0511 2 in such command lists do not prompt for topics or subtopics.
0512 2
0513 2 CIS_PTR = .DBG$GL_CISHEAD;
0514 2 IF .CIS_PTR[CISC$B_INPUT_TYPE] EQL CIS_INPBUF
0515 2 THEN
0516 2 CIS_PTR = .CIS_PTR[CISC$A_NEXT_LINK];
0517 2
0518 2 IF (.CIS_PTR[CISC$B_INPUT_TYPE] NEQ CIS_DBG$INPUT) AND
0519 2 (.CIS_PTR[CISC$B_INPUT_TYPE] NEQ CIS_RAB)
0520 2 THEN
0521 2 BEGIN
0522 2 GET_STR[DSC$W_LENGTH] = 0;
0523 2 RETURN SSS_NORMAL;
0524 2 END;
0525 2
0526 2
```

```

399 0527 2 ! We are reading from the user's input terminal (SYSS$INPUT or DBG$INPUT)
400 0528 2 ! or we are reading from an indirect command file. If we are reading from
401 0529 2 ! the terminal and we are in screen mode, we update the screen at this
402 0530 2 ! point so that the user sees his current HELP output before being prompted
403 0531 2 ! for another topic or subtopic.
404 0532 2
405 0533 2 IF .DBG$GL_SCREEN_MODE AND
406 0534 2 (.CIS_PTR[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
407 0535 2 THEN
408 0536 2     DBG$SCR_OUTPUT_SCREEN();
409 0537 2
410 0538 2
411 0539 2 ! If keypad input mode is enabled and we are reading from the terminal, we
412 0540 2 ! read a line of input by calling the keypad input routine.
413 0541 2
414 0542 2 IF .DBG$GB_KEYPAD_INPUT AND
415 0543 2 (.CIS_PTR[CIS$B_INPUT_TYPE] EQL CIS_DBG$INPUT)
416 0544 2 THEN
417 0545 2     BEGIN
418 0546 2         STATUS = SMG$READ_COMPOSED_LINE(DBG$GL_KEYBOARD_ID,
419 0547 2             DBG$GL_KEY_TABLE_ID,
420 0548 2             .GET_STR,
421 0549 2             .PROMPT_STR,
422 0550 2
423 0551 2             *** Note - the fifth parameter (DEFAULT_STATE) is being removed from
424 0552 2             *** this routine, according to Steve Lionel. If Steve's change doesn't
425 0553 2             *** make it this build, however, the '0' must be restored here.
426 0554 2
427 0555 2             0,
428 0556 2
429 0557 2             LENGTH);
430 0558 2     GET_STR[DSC$W_LENGTH] = .LENGTH;
431 0559 2     IF .STATUS EQL SMG$_EOF THEN STATUS = RMS$_EOF;
432 0560 2     RETURN .STATUS;
433 0561 2     END;
434 0562 2
435 0563 2
436 0564 2 ! We are either reading from the user's terminal in the normal way using
437 0565 2 ! RMS or we are reading from an indirect command file. Hence we set up
438 0566 2 ! the RAB and call RMS to give the prompt and read a line.
439 0567 2
440 0568 2     DBG$GL_INPRAB[RAB$W_USZ] = .GET_STR[DSC$W_LENGTH];
441 0569 2     DBG$GL_INPRAB[RAB$L_UBF] = .GET_STR[DSC$A_POINTER];
442 0570 2     DBG$GL_INPRAB[RAB$B_PSZ] = .PROMPT_STR[DSC$W_LENGTH];
443 0571 2     DBG$GL_INPRAB[RAB$L_PBF] = .PROMPT_STR[DSC$A_POINTER];
444 0572 2     STATUS = $GET(RAB = DBG$GL_INPRAB);
445 0573 2
446 0574 2
447 0575 2 ! Put the number of characters read back into the string descriptor. Then
448 0576 2 ! return with the status we got back from $GET.
449 0577 2
450 0578 2     GET_STR[DSC$W_LENGTH] = .DBG$GL_INPRAB[RAB$W_RSZ];
451 0579 2     RETURN .STATUS;
452 0580 2
453 0581 2 END;

```

## .EXTRN SYSSGET

000C 00000 INPUT\_HELP LINE:

53	00000000G	00	9E	00002	.WORD	Save R2, R3	: 0470
5E		04	C2	00009	MOVAB	DBG\$GL_INPRAB+32, R3	
50	00000000G	00	D0	0000C	SUBL2	#4, SP	
02	02	A0	91	00013	MOVL	DBG\$GL_CISHEAD, CIS_PTR	: 0513
		04	12	00017	CMPB	2(CIS_PTR), #2	: 0514
50	08	A0	D0	00019	BNEQ	1\$	
52	02	A0	9A	0001D	MOVL	8(CIS_PTR), CIS_PTR	: 0516
		0C	13	00021	MOVZBL	2(CIS_PTR), R2	: 0518
01		52	91	00023	BEQL	2\$	
		07	13	00026	CMPB	R2, #1	: 0519
	04	BC	B4	00028	BEQL	2\$	
50		01	D0	0002B	CLRW	@GET_STR	: 0522
		04	0002E	MOVL	#1, R0		: 0523
0B	00000000G	00	E9	0002F	RET		
		52	D5	00036	BLBC	DBG\$GL_SCREEN_MODE, 3\$	: 0533
		07	12	00038	TSTL	R2	: 0534
00000000G	00	00	FB	0003A	BNEQ	3\$	
32	00000000G	00	E9	00041	CALLS	#0, DBG\$SCR_OUTPUT_SCREEN	: 0536
		52	D5	00048	BLBC	DBG\$GB_KEYPAD_INPUT, 4\$	: 0542
		2E	12	0004A	TSTL	R2	: 0543
		5E	DD	0004C	BNEQ	4\$	
7E	04	AC	7D	0004E	PUSHL	SP	: 0546
	00000000G	00	9F	00052	MOVQ	GET_STR, -(SP)	: 0548
	00000000G	00	9F	00058	PUSHAB	DBG\$GL_KEY_TABLE_ID	: 0546
00000000G	00	05	FB	0005E	PUSHAB	DBG\$GL_KEYBOARD_ID	
04	BC	6E	B0	00065	CALLS	#5, SMG\$READ_COMPOSED_LINE	
00000000G	8F	50	D1	00069	MOVW	LENGTH, @GET_STR	: 0558
		2F	12	00070	CPL	STATUS, #SMG\$_EOF	: 0559
50	0001827A	8F	D0	00072	BNEQ	5\$	
		04	00079	MOVL	#98938, STATUS		
52	04	AC	D0	0007A	RET		: 0560
63		62	B0	0007E	MOVL	GET_STR, R2	: 0568
04	A3	A2	D0	00081	MOVW	(R2), DBG\$GL_INPRAB+32	
51	08	AC	D0	00086	MOVL	4(R2), DBG\$GL_INPRAB+36	: 0569
14	A3	61	90	0008A	MOVL	PROMPT_STR, RT	: 0570
10	A3	A1	D0	0008E	MOVB	(R1), DBG\$GL_INPRAB+52	
		E0	A3	9F	MOVL	4(R1), DBG\$GL_INPRAB+48	: 0571
00000000G	00	01	FB	00096	PUSHAB	DBG\$GL_INPRAB	: 0572
	62	02	A3	B0	CALLS	#1, SYSSGET	
		04	000A1	5\$:	MOVW	DBG\$GL_INPRAB+34, (R2)	: 0578
					RET		: 0581

; Routine Size: 162 bytes, Routine Base: DBG\$CODE + 0178

: 454 0582 1  
: 455 0583 0 END ELUDOM

.EXTRN LIB\$SIGNAL

# PSECT SUMMARY

Name	Bytes	Attributes
DBG\$CODE	538	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)
DBG\$PLIT	44	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(0)

## Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	19	0	1000	00:01.8
-\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
-\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	48	3	97	00:02.1
-\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	0	0	31	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	2	0	22	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	0	0	12	00:00.3

## COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGNHLP/OBJ=OBJ\$:DBGNHLP MSRC\$:DBGNHLP/UPDATE=(ENH\$:DBGNHLP)

Size: 538 code + 44 data bytes  
Run Time: 00:16.5  
Elapsed Time: 00:56.3  
Lines/CPU Min: 2126  
Lexemes/CPU-Min: 12401  
Memory Used: 137 pages  
Compilation Complete

0087 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

DBGNMSG  
LIS

DBGNHELP  
LIS

DBGNPARSE  
LIS

DBGNEXITE  
LIS

DBGNPNP  
LIS